MIP Program Written by: Saam Ostovari
11/15/2013
NDA SD2013-802

MIP_Cpp_v3 Basic Usage Instructions

The following program is written to be a general program that can be used for any control and balancing of MIP or similar inverted pendulum vehicle using the Arduino microcontroller. This program specifically incorporates code for use with any analog IMU, the MPU6050 digital IMU, quadrature encoders, JY-MCU bluetooth board and dual motor drivers. To be able to use this on different robots minor modifications need to be made due to different robot configurations and pin usage.

Before anything can be done with the code, a few things need to be done:

1) Load the bluetooth library into the Arduino libraries folder
    a. The bluetooh library can be found in the "Libraries to Add" folder
2) (IF USING DIGITAL IMU) Change the I2C frequency for the arduino
    a. Locate and open the following file in your Arduino folder: .../libraries/Wire/utility/twi.h.
    b. Located the following bit of code:
       ```
       #ifndef TWI_FREQ
       #define TWI_FREQ 100000L
       #endif
       ```
    c. Change the 100000L to 400000L
       ```
       #ifndef TWI_FREQ
       #define TWI_FREQ 400000L
       #endif
       ```
    d. Save and close

You may now start the arduino environment. If you already had it open then you will need to restart to allow the changes just made to take effect.

1) Open the file labeled MIP_Cpp_v3
2) You will see the main file open with plenty of tabs for the other functions. We will go through each one
3) In tab: MIP_Cpp_v3
    a. Under the void setup() loop you will need to change a few things that you initialize
        i. If you are using a digital IMU uncomment Initialize_Digital_IMU() and comment Initialize_IMU(). If you are using an Analog IMU do the opposite.
        ii. If you have a digital pin that is connected to the arduino's reset pin then uncomment Initialize_Arduino_Hard_Reset(). Otherwise, leave it commented
        iii. If you are using bluetooth then uncomment Initialize_Bluetooth()
        iv. If you would like to have quiet motors then uncomment Initialize_Faster_PWM_Freq(). Note: by uncommenting this function time will not display correctly. In other words, the micros() function and other similar functions will not work as expected.
        v. Select what speed you'd like your timer interrupt to run at by uncommenting the appropriate function: Initialize_?ms_Timer_Interrupt()
    b. Under the ISR function

        i. Comment the safety checks if you do not want to have any safeties such as fall detection in your robot. This is recommended for when you are first testing out a new robot.
        ii. Select which controller you are using by uncomment one of the following functions: SLC_Control() or LQR_Control()
    c. Under the main void loop()
        i. MPU_READ() will need to be uncommented if you are using a digital IMU. Otherwise keep it commented.

4) In tab: Bluetooth
    a. Depending on driving desirability you can change the following two variables:
    float scaleTurn = ?;
    float scaleForward =?;

5) In tab Analog_IMU
    a. IF USING DIGITAL IMU, **<u>COMMENT ALL</u>** CODE and move to next tab
    b. You will need to determine the voltage going into your analog sensors and change Vref accordingly
    c. If using Analog IMU, start by checking Gyro and Accelerometer sensitivity values.
    d. Determine what your Sensor Zero Gravity Voltage are. You need to do this for both the gyro and the accelerometer
    e. Make adjustments to the xAccelDir, zAccelDir and yGyroDir depending on how your sensor is oriented in the robot
    f. You will need to adjust the AccelOffset after you get the robot running but for now set it to 0.
    g. Depending on how you orient your sensor axis and how you set the direction signs (+or-) you may need to alter the atan2(?,?) function. If you need to do this then you will need to locate the variable acc_theta under both the Initialize_Digital_IMU() and the IMU_Update() functions. Change it in both places.

6) In tab: Digital IMU
    a. IF USING ANALOG IMU, **<u>COMMENT ALL</u>** CODE and move to next tab
    b. If using digital IMU, start by checking Gyro and Accelerometer sensitivity values.
    c. Make adjustments to the xAccelDir, zAccelDir and yGyroDir depending on how your sensor is oriented in the robot
    d. You will need to adjust the AccelOffset after you get the robot running but for now set it to 0.
    e. Depending on how you orient your sensor axis and how you set the direction signs (+or-) you may need to alter the atan2(?,?) function. If you need to do this then you will need to locate the variable acc_theta under both the Initialize_Digital_IMU() and the IMU_Update() functions. Change it in both places.

7) In tab: Encoder
    a. Make sure that your encoder pins are defined correctly, otherwise correct it
    b. Make the appropriate changes to the following three variables:
        - const float CountsPerRev = 200;
        - const float CountsPerTick = 2;
        - const float GearRatio = 1;

8) In tab: Estimator
    a. Nothing to change

9) In tab: FallDetection
    a. Nothing to change

10) In tab: MPU6050dav.h
    a. Nothing to change
11) In tab: RC_Drive
    a. Make sure your motor pins are defined correctly
    b. Change the MaxPWM to the value you'd like (I never like giving full power to my motors). If you are not sure what this is then just change it to 255
12) In Tab: SLC_Controller
    a. Provide the appropriate equations/constants for the controller you are using
13) In tab: Safeties
    a. Under Safety_Checks() function you can comment out specific safeties you'd like to leave out instead of disable all safeties as done in section 3.b.i
14) In tab: Timer Interrupt
    a. Nothing to change

Above outlined the important steps for using the provided code on a MIP platform.

Disclaimer:
Do not change the names of the tabs as this will cause issues with the code. The Arduino IDE compiles the code in alphabetical order after the main program. So changing that order could break the code.